Bioinformatics 30.12 (2014): 1759-1761

DOI: http://dx.doi.org/10.1093/bioinformatics/btu099

ADaCGH2: parallelized analysis of (big) CNA data

Ramon Diaz-Uriarte*

This is a pre-copyedited, author-produced PDF of an article accepted for publication in *Bioinformatics* following peer review. The version of record Bioinformatics (2014) 30 (12): 1759-1761. doi: 10.1093/bioinformatics/btu099 is available online at: `http://bioinformatics.oxfordjournals.org/content/30/12/1759`

Abstract

1 Motivation:

Studies of genomic DNA copy number alteration (CNA) can deal with data sets with several million probes and thousands of subjects. Analyzing these data with currently available software (e.g., as available from BioConductor) can be extremely slow and might not be feasible because of memory requirements.

2 Results:

We have developed a BioConductor package, ADaCGH2, that parallelizes the main segmentation algorithms (using forking on multicore computers or parallelization via MPI, etc, in clusters of computers), and uses *ff* objects for reading and data storage. We show examples with data of 6 million probes per array; we can analyze data that would otherwise not fit in memory, and compared to the non-parallelized versions we can achieve speed ups of 25 to 40 times on a 64-cores machine.

3 Availability:

ADaCGH2 is an R package available from BioConductor. Version 2.3.11 or higher is available from the development branch: `http://www.bioconductor.org/packages/devel/bioc/html/ADaCGH2.html`.

*Department of Biochemistry, Universidad Autónoma de Madrid, Instituto de Investigaciones Biomédicas "Alberto Sols" (UAM-CSIC), Madrid, Spain

4 Contact:

ramon.diaz@iib.uam.es

5 Introduction

Current studies of genomic copy number alterations (CNA) are using platforms with several million probes per array and several thousand subjects (e.g., Grozeva *et al.*, 2012) but the canonical implementations of the widely used, open source packages for the analysis of CNA data did not allow for parallelized execution of the analysis. This makes it difficult to use clusters of servers, and does not take advantage of the trends in parallel computing towards multicore machines (servers with 16 to 124 cores or laptops with two or four cores are nowadays common). Moreover, and especially for R/BioConductor software, the available packages will not be able to analyze big data sets if these are larger than about a quarter to a half of the available memory (unless one uses time-consuming, and *ad hoc*, manual partition of the input and subsequent recombination of the output — see discussion in section "Why ADaCGH2 instead of a 'manual' solution" in the vignette of the package).

Here I describe ADaCGH2, a BioConductor package designed to address the above deficiencies. I leverage on two R packages, `parallel`, part of the standard set of R packages, and `ff` (Adler *et al.*, 2013), and combine them, to provide:

- Parallelized analysis. I have parallelized the most widely used segmentation approaches that can be applied to CNA data, including both CGH and SNP arrays (Valsesia *et al.*, 2013) —but also covering sequencing data, when these have been appropriately processed, but see Duan *et al.* (2013), Zhao *et al.* (2013), Wu *et al.* (2013), Zheng *et al.* (2013) . The methods available are CBS (Venkatraman and Olshen, 2007), HaarSeg (Ben-Yaacov and Eldar, 2008), HMM (Fridlyand *et al.*, 2004), BioHMM (Marioni *et al.*, 2006), the Wavelet-based method from Hsu *et al.* (2005), GLAD (Hupe *et al.*, 2004), and CGHseg (Picard *et al.*, 2005), and two merging algorithms. Some of those methods, however, are not suitable for very large data sets —see details in section 1.2.1 of the "benchmarks.pdf" package vignette .

  I use package `parallel` to provide parallelization using: a) forking, for single multicore computers; b) parallelization with MPI, sockets, PVM, etc, for clusters built of several computers.

- The ability to analyze data that cannot fit in memory. Using `ff` we only access the section of the data currently being analyzed, keeping

in RAM and moving between processes only a pointer to the rest of the data on disk.

- Parallelization of data input and output and plotting.
- Input from, and output to, other BioConductor packages.

Here I present the main functions of the package, the differences with former version , and some benchmarks. A full set of examples, further benchmarks, and detailed suggestions for usage , are included in the package vignettes.

6 Differences with the former version

ADaCGH was first developed to provide parallelized analysis of CNA data for web-based applications (Carro *et al.*, 2010; Diaz-Uriarte and Rueda, 2007). The first version parallelized eight segmentation algorithms (using MPI), was available from CRAN, and was last updated on 2009, but will no longer run without tweaks as it depends on a package, papply, that will not install in versions of R from several years ago. Next, parallelization was extended so clusters were not limited to MPI clusters, and *ff* objects were used for storage; that version is available as v. 1.10, from BioConductor 2.12. For the current version most of the code has been rewritten to use forking, data handling and reading of input data has been completely modified so that data much larger than available memory can be read and analyzed, and missing value handling has been changed to use all available data per array . The vignette `benchmarks.pdf` provides extensive comparisons between the new ($\geq$ 2.3.5) and latest previous running versions (v. 1.10), but the main differences between these two versions are:

- **Reading and analysis of large data sets** The new version can read data sets much larger than the old one and, in fact, data sets much larger than available memory (see details in section 7). As a consequence of being able to read much larger data sets, the new version can analyze data sets much larger than the old one.
- **Missing value handling** The old version used row-wise deletion of missing values when reading data (i.e., a probe would be deleted from the data if it had one missing value in any array/column). The new version deals with missing values array by array, so for each array (or column) all available data (or probes) are used in the segmentation.
- **Forking and clusters** The new version of ADaCGH2 allows for the usage of forking or an explicit cluster (e.g., MPI, sockets, etc) to parallelize reading and analysis. In POSIX operating systems (including

Unix, GNU/Linux, and Mac OS), forking can be faster, less memory consuming, and much easier to use than a cluster.

- **Flexibility of reading data and compatibility with former version** The new version of ADaCGH2 has not removed the mechanisms of reading data available in the old version (when data are small or memory is plentiful, reading data from a single RData is an available option) and accepts data read by the former version. However, the old version cannot accept data read by the new version as it assumes that data that have been read contain no missing values.

These differences in implementation, however, do not affect the underlying core code for the algorithms, which is the same as in the previous version. There have been, however, changes in some defaults, to adapt the package to really large data (e.g., using MAD as merging default or using "haarseg" as the "smoothfunc" for daglad, following recommendations in the package vignette for GLAD).

7 Benchmarks

Figure 1 shows benchmarks of reading and analyzing data with 6,067,433 probes per array/column. Those figures compare memory usage and wall time of the old and new versions and of the non-parallelized versions in two different machines (data for the figures, as well as benchmarks for a third machine, and with MPI over two machines, are available from the vignette "benchmarks.pdf"). To give an idea of sizes, the RData file for the 1000 arrays data is of about 41 GB and the directory with the data for 2000 columns/arrays occupies about 198 GB.

Compared to the non-parallelized version, in the analysis of data ADaCGH2 leads to speed increases by factors of 25 to 40 times in the 64 cores machines and 7 to 10 times in the 12 cores machines , and allows us to analyze data that would not fit in memory .

Compared to the former version, the new version uses less memory for analysis. More important, the new version allows us to read and analyze much larger data sets. In the 256 and 384 GB machines the old version cannot read data sets with 2000 or more arrays (R runs out of memory) and in the machine with 64 GB of RAM it cannot read data with 500 or more arrays (R runs out of memory); as can be seen from the figure, the old version shows a steep linear increase in memory consumption with number of arrays. In sharp contrast, with the new version we can read and analyze 4000 arrays in a machine with only 64 GB of RAM (see Figure 1 b) and the scaling of memory usage with number of arrays suggests that much larger data sets could be read and analyzed. In addition, we can obtain speed ups

by factors of 2x to 10x (depending on machine and number of arrays) in the reading step as it is parallelized.

8 Work flow

Figure 2 shows the usual sequence of calls with ADaCGH2. `inputToADaCGH` accepts input in different formats, including objects used by limma (Smyth, 2005) and snapCGH (Smith *et al.*, 2009), and produces R data frames or *ff* objects, after performing several checks and data sanitation. If data are read from a directory with one-column files reading is parallelized (`cutFile` allows splitting a text file into one-column files). `pSegment` can take as input R data frames and *ff* objects produced by `inputToADaCGH`. `pSegment` can use multiple cores or multiple computers and it can accept as input data frames or *ff* objects; when running on a cluster only *ff* objects are used (to avoid passing around large objects and to allow analyzing large data sets). The output from `pSegment` can be converted so it is accepted by the CGHregions package (Vosse and van de Wiel, 2009), and creation of figures is also parallelized. Note than

9 Conclusion

ADaCGH2 should be of immediate use for researchers involved in the analysis of CNA data. Parallelization allows it to speed up data processing , and it can handle data that will not fit in memory with excellent scaling of memory usage with number of arrays. These behaviors are needed for the analyses of platforms with increasing number of probes and multi-center studies with thousands of subjects.

Acknowledgement

O. M. Rueda and D. Rico for collaboration in previous versions of ADaCGH, and help debugging under Mac OS. O. M. Rueda for the initial benchmarking data. C. Lázaro-Perea, O. M. Rueda, I. López, and three anonymous reviewers for comments on the ms.

Funding Project BIO2009-12458 (from the Spanish MINECO). This paper is also a contribution from project IMCOMP, CEMU-2013-14, of the Universidad Autónoma de Madrid.

References

Adler, D., Glser, C., Nenadic, O., Oehlschlgel, J., and Zucchini, W. (2013). *ff: memory-efficient storage of large data on disk and fast access functions*. R package version 2.2-11.

Ben-Yaacov, E. and Eldar, Y. C. (2008). A fast and flexible method for the segmentation of aCGH data. *Bioinformatics (Oxford, England)*, **24**(16), i139–i145.

Carro, A., Rico, D., Rueda, O. M., Diaz-Uriarte, R., and Pisano, D. G. (2010). waviCGH: a web application for the analysis and visualization of genomic copy number alterations. *Nucleic acids research*, **38 Suppl**, W182–7.

Diaz-Uriarte, R. and Rueda, O. M. (2007). ADaCGH: A parallelized web-based application and R package for the analysis of aCGH data. *PloS one*, **2**(1), e737.

Duan, J., Zhang, J.-G., Deng, H.-W., and Wang, Y.-P. (2013). Comparative studies of copy number variation detection methods for next-generation sequencing technologies. *PloS one*, **8**(3), e59128.

Fridlyand, J., Snijders, A. M., Pinkel, D., and Albertson, D. G. (2004). Hidden Markov models approach to the analysis of array CGH data. . *Journal of Multivariate Analysis*, **90**, 132–153.

Grozeva, D., Conrad, D. F., Barnes, C. P., Hurles, M., Owen, M. J., O'Donovan, M. C., Craddock, N., and Kirov, G. (2012). Independent estimation of the frequency of rare CNVs in the UK population confirms their role in schizophrenia. *Schizophrenia research*, **135**(1-3), 1–7.

Hsu, L., Self, S. G., Grove, D., Randolph, T., Wang, K., Delrow, J. J., Loo, L., and Porter, P. (2005). Denoising array-based comparative genomic hybridization data using wavelets. *Biostatistics*, **6**(2), 211–226.

Hupe, P., Stransky, N., Thiery, J. P., Radvanyi, F., and Barillot, E. (2004). Analysis of array cgh data: from signal ratio to gain and loss of dna regions. *Bioinformatics (Oxford, England)*, **20**(20), 3413–3422.

Marioni, J. C., Thorne, N. P., and Tavaré, S. (2006). BioHMM: a heterogeneous hidden Markov model for segmenting array CGH data. *Bioinformatics*, **22**(9), 1144–1146.

Picard, F., Robin, S., Lavielle, M., Vaisse, C., and Daudin, J. J. (2005). A statistical approach for array CGH data analysis. *BMC Bioinformatics*, **6**.

Smith, M. L., Marioni, J. C., McKinney, S., Hardcastle, T., and Thorne, N. P. (2009). *snapCGH: Segmentation, normalisation and processing of aCGH data.* R package version 1.31.0.

Smyth, G. K. (2005). Limma: linear models for microarray data. In R. Gentleman, V. Carey, S. Dudoit, R. Irizarry, and W. Huber, editors, *Bioinformatics and Computational Biology Solutions Using R and Bioconductor*, pages 397–420. Springer, New York.

Valsesia, A., Macé, A., Jacquemont, S., Beckmann, J. S., and Kutalik, Z. (2013). The Growing Importance of CNVs: New Insights for Detection and Clinical Interpretation. *Frontiers in Genetics*, **4**(May), 1–19.

Venkatraman, E. S. and Olshen, A. B. (2007). A faster circular binary segmentation algorithm for the analysis of array CGH data. *Bioinformatics (Oxford, England)*, **23**(6), 657–663.

Vosse, S. and van de Wiel, M. (2009). *CGHregions: Dimension Reduction for Array CGH Data with Minimal Information Loss.* R package version 1.7.1.

Wu, Y., Tian, L., Piratsu, M., Stambolian, D., Li, H. (2013). MATCHCLIP: locate precise breakpoints for copy number variation using CIGAR string by matching soft clipped reads. *Frontiers in Genetics*, **4** (157), 1–7.

Zhao, M., Wang, Q., Wang, Q., Jia, P., and Zhao, Z. (2013). Computational tools for copy number variation (CNV) detection using next-generation sequencing data: features and perspectives. *BMC Bioinformatics*, **14**(Suppl 11), S1.

Zheng, C., Miao, X., Li, Y., Huang, Y., Ruan, J., Ma, X., Wang, L., Wu, C., Cai, J. (2013). Determination of genomic copy number alteration emphasizing a restriction site-based strategy of genome re-sequencing. *Bioinformatics (Oxford, England)*, **29**(22), 2813–2821.]

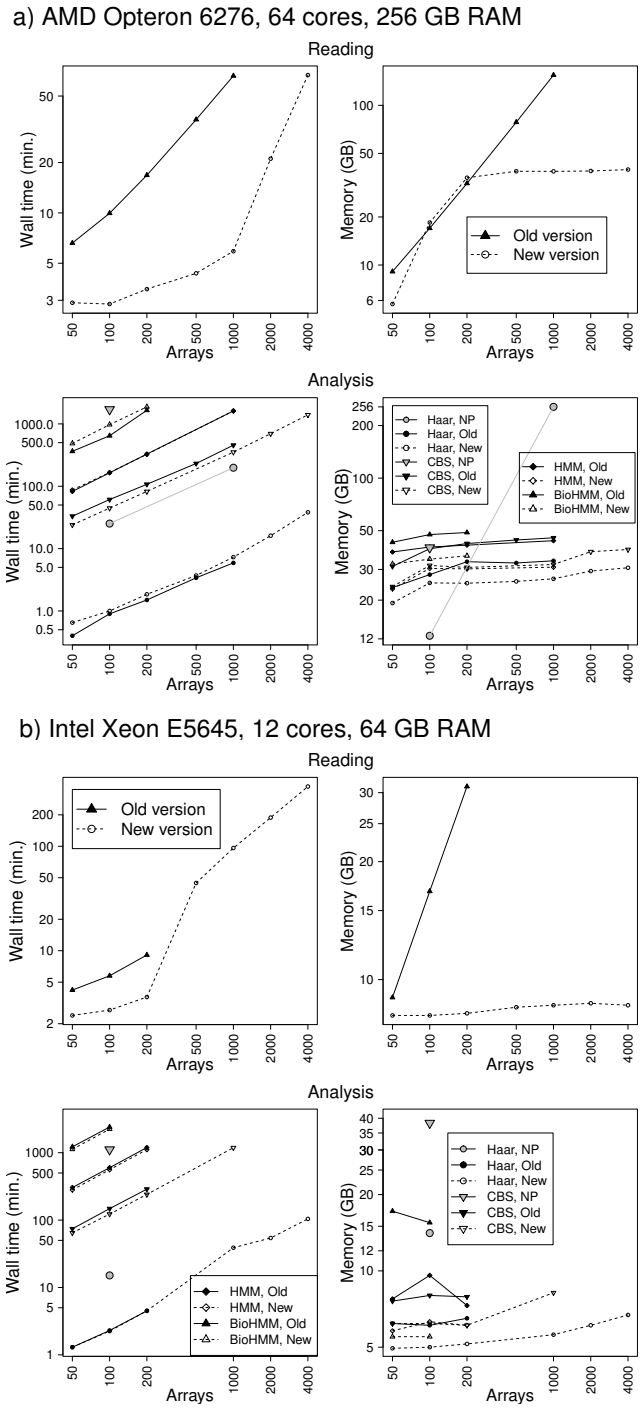

Figure 1: Wall time and memory use (summed over all spawned processes) of reading and analysis as a function of number of arrays. Reading: comparison between new and old versions. Analysis: new and old versions with four segmentation methods, and non-parallelized (NP) for two methods. No benchmark allowed to run for more than 36 hours. Without parallelization, in the AMD machine no runs of CBS with 1000 arrays or HaarSeg with 2000 can be done (R runs out of memory); in the Intel machine no runs for 1000 arrays with any method can be done (R runs out of memory).

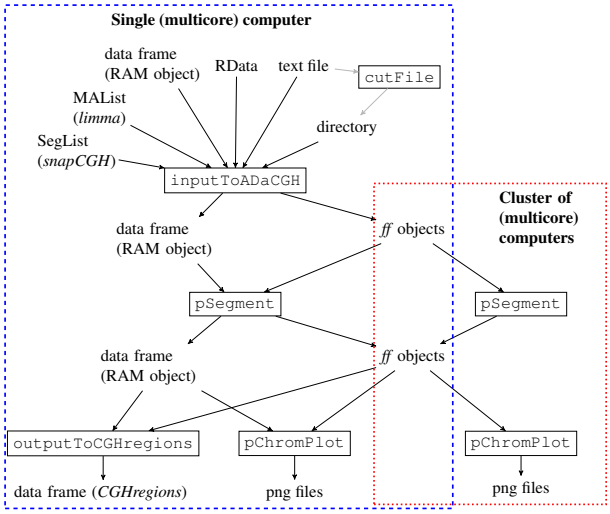

Figure 2: Work flow from input data to figures. R functions shown with courier font inside boxes. In italics names of other packages which can provide input/accept output. Data frames are stored in memory, in contrast to *ff* objects. Data input and conversion to *ff* objects is done with `inputToADaCGH` (maybe after using `cutFile` for parallelized reading of single-column files). Segmentation is carried out with `pSegment`, and results can then be plotted (`pChromPlot`) or used by other packages (`outputToCGHregions`). When using *ff* objects, after data input (in a single machine) the remaining analyses can be conducted in a cluster.